(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 January 2001 (11.01.2001)

PCT

(10) International Publication Number
WO 01/03308 A1

(51) International Patent Classification[7]: H03M 13/23

(21) International Application Number: PCT/EP99/04725

(22) International Filing Date: 6 July 1999 (06.07.1999)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: FRAUNHOFER-GESELLSCHAFT ZUR FÖRDERUNG DER ANGEWANDTEN FORSCHUNG E.V. [DE/DE]; Leonrodstrasse 54, D-80636 München (DE).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: LEYH, Martin [DE/DE]; Drausnickstrasse 153, D-91052 Erlangen (DE). SPEITEL, Martin [DE/DE]; Herrmann Löns-Strasse 7, D-92318 Neumarkt (DE). KÖHLER, Stefan [DE/DE]; Schulstrasse 1, D-96472 Rödental (DE).

(74) Agent: SCHOPPE, Fritz; SCHOPPE, ZIMMERMANN & STÖCKELER, POSTFACH 71 08 67, D-81458 München (DE).

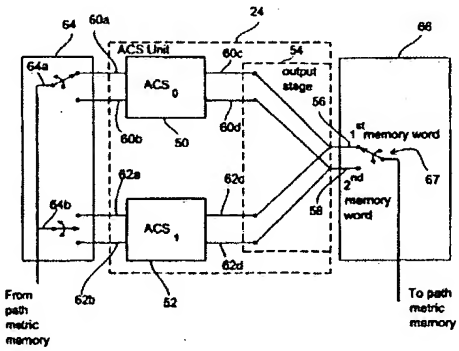
(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: VITERBI DECODER



(57) Abstract: A scalable Viterbi decoder comprises a transition metrics calculation unit (22), an add-compare-select unit (24), a survivor memory unit (28), a path metrics memory (26), and a path metrics memory address generator. The add-compare-select unit (24) comprises at least two add-compare-select devices each add-compare-select device being arranged for processing at least two trellis butterflies in at least two subsequent sub-cycles. The path metrics memory (26) comprises individually addressable memory cells, the size of each cell being such that a number of path metrics can be stored in one cell, the number of path metrics stored in one cell being equal to the number of add-compare-select devices. The path metrics memory address generator is arranged for generating write addresses which are identical to the read addresses, and for generating read addresses for the subsequent cycle by left-rotating the bits of read addresses of the preceding cycle. The add-compare-select unit further comprises an output stage for combining survivor path metrics output by the plurality of add-compare-select devices such that each memory cell receives one path metrics of each add-compare-select device per sub-cycle, and that one memory cell receives the path metrics required for the processing of one trellis butterfly. The scalable Viterbi decoder architecture allows the deployment of area-efficient and power-efficient random access memories with scalable address and data width. Adaption to different data rates becomes easily feasible.

WO 01/03308 A1

Viterbi Decoder

Specification

The present invention relates to Viterbi decoders and, in particular, to scalable Viterbi decoders, i.e., Viterbi decoders that perform serial processing as well as parallel processing in one working cycle.

The Viterbi algorithm is an important decoding method in digital communications. The Viterbi algorithm is an optimal Maximum-Likelihood Sequence Estimator. The Viterbi algorithm is applied, e.g., for channel decoding of convolutional codes, channel estimation, pattern recognition, etc.

Because of its importance, intensive research has been conducted in the past on an efficient VLSI ("VLSI = very large scale integration") implementation of Viterbi decoders. One of the main objectives is an area-optimized implementation for given performance requirements, e.g., data rate and the number of states of the Viterbi decoder. Of special interest are scalable and parameterizable Viterbi decoder architectures which are adapatable to different applications. In general, Viterbi decoders can be implemented by using a fully parallel architecture or a fully serial architecture or a mixed architecture including parallel and serial portions. The parallel architecture will yield the highest performance in terms of speed, whereas the serial architecture will result in the lowest performance in terms of speed. However, the fully parallel architecture will lead to Viterbi decoder chips that consume much power and chip area. To the contrary, the fully serial architecture results in a Viterbi decoder chip that uses less power and chip area.

A scalable and parameterizable Viterbi architecture is an architecture that results in speed, power consumption and

area consumption parameters that lie between the two extremes. As it has been outlined above, Viterbi decoders have become key elements in a broad variety of different applications. Therefore, it is not convenient to demand identical decoder parameters, such as speed, area consumption and power consumption, for all Viterbi decoder applications. In contrast thereto, the parameter requirements change from application to application. This leads to scalable and parameterizable decoder architectures, i.e., decoder architectures that include serial processing portions as well as parallel processing portions.

The bottleneck of the Viterbi algorithm is the data-dependent and, therefore, non-linear feedback in the Add-Compare-Select (ACS) computations which determines the achievable data throughput of all Viterbi decoder architectures. The ACS computers perform the computation for one butterfly in the trellis diagram, i.e., for 2 states in parallel.

The highest data throughput can be achieved by fully parallel Viterbi decoder architectures which have one ACS computer for each trellis butterfly. The lowest data throughput is achieved by serial architectures which perform all ACS computations serially with a single ACS computer, as is the case, e.g., in digital signal processors (DSPs).

In the following, a brief summary of the Viterbi algorithm and Viterbi decoders in general is given. This summary is based on "The Viterbi Algorithm", G.D. Forney, Proceedings of the IEEE, Vol. 61, No. 3, March 1973.

In Fig. 1, a general structure of a convolutional coder is given. A convolutional coder generates a coded signal, which can be decoded by a Viterbi decoder. The convolutional coder, in general, consists of an input stage 10a and an arbitrarily selected number of memory stages 10b, 10c. The stages have a length of k bits, such that each stage is able

to hold a k-ary information symbol. This fact is illustrated in Fig. 1 by the notations of the input line and the lines connecting the different stages. These notations indicates that these lines are parallel lines including k individual serial lines. In the first cycle, a first information symbol having k bits is loaded into the input stage 10a. In another cycle, the information symbol in the input stage 10a is shifted out into the first memory stage 10b. Simultaneously, another information symbol is loaded into the input stage 10a. In the next cycle, the information symbol located in the memory stage 10b is shifted into the second memory stage 10c. Additionally, the information symbol located in the input stage 10a is shifted out and loaded into the first memory stage 10b. Finally, a new information symbol is loaded into the input stage 10a. The individual bits of the symbols are combined by, e. g., modulo-2 adders 12a, 12b and 12c. At the output of the modulo-2 adders 12a to 12c, a coded symbol sequence is generated. In general, a convolutional code involves that, in the coded symbol sequence, the information contents relating to the number of bits is lower than in the information bits input into the input stage. In other words, a convolutional code adds redundancy to an input signal, which redundancy is used, for example, for error recovering measures, as is the case in digital communications. Thus, it can be stated that a convolutional coder can be described by the term "code rate", which code rate is determined by the number of input bits, which is k, divided by the number of output bits, which is n. In general, it can be stated that n is greater than k. Another term relating to a convolutional coder is the constraint length L, which is the number of stages, each stage having a length corresponding to the k bits.

The shift-register process for the generation of the convolutional code can be modelled as Markov process or as a Finite State Machine (FSM), respectively, since the code bits, i.e., the output data, are dependent only on the information bits, i.e., the input data and the current

content of the shift register, which is called the shift register state. The state of the convolutional code, i.e., the current content of the shift register, corresponds to the k multiplied by L bits in the memory stages. Accordingly, the number, S, of states of the shift register is defined by:

$$S = 2^{k(L-1)} \qquad \text{(Equation 1)}$$

The number, B, of possible states transitions is defined as follows:

$$B = 2^{kL} \qquad \text{(Equation 2)}$$

In Fig. 3, reference is made to a special convolutional code which can be generated by a shift register comprising one input stage 10a and three memory stages 10b to 10d. In this example, k is set to 1. The shift register stages are combined by two combiners 12a and 12b. Thus, it can be seen from the upper portion of Fig. 3 that one bit input into the input stage 10a results in two output bits c_1, c_2. Thus, a convolutional coder shown in the upper portion of Fig. 3 has a code rate of 1/2 and a constraint length L of 4. In the upper portion of Fig. 3, the so-called trellis diagram of such a coder is shown. To the left-hand side, the states of the shift register composed of the stages 10a to 10d are numbered in their natural order by means of binary numbers. Thus, the lowest state in Fig. 3 is denoted as state 000, whereas the highest state is denoted as state 111.

The trellis diagram has 8 states and 16 possible state transitions. For each discrete time instant (the time axis is drawn in the horizontal direction), all states are depicted as nodes in according columns and all possible state transitions are depicted as branches between the nodes. On each branch, the resulting code bits of the state transitions are shown. The notation associated with the individual branches includes the new received information

bit followed by a vertical line (|). On the right-hand side of the vertical line (|), the two bits c_1 and c_2 of the coded signal are shown. The branch from the state 000 at a time instant i to the state 000 at the time instant i+1 has the notation 0|00, which means that this branch is valid for a new received input bit which is 0, and for current code bits c_1 and c_2 having values of 00. Another approach to the state transitions given in the trellis diagram of Fig. 3 is as follows. When, for example, the shift register state 100 is considered, this shift register state will change to a shift register state of 010, when a 0 bit is shifted into the shift register. On the other hand, the shift register state 100 is changed to a shift register state 110, when a bit having a value of 1 is shifted into the shift register. Thus, is can be seen that each shift register state can be followed by only two shift register states out of the 8 states, depending on the new input bit, which can have a value of 0 or 1, only. The trellis diagram is used for determining the actual state that the shift register in the coder had when a certain information bit was coded. In other words, it can be said that a Viterbi decoder tracks all state transitions from all states to the respective following states, wherein each branch, i.e., each state transition, has a certain transition metric assigned to it.

It becomes clear from the following discussion that each pair of states in the trellis diagram has another pair of states assigned to it that can be reached when a new information bit is decoded. Such two input states and their corresponding output states are called a trellis butterfly. One trellis butterfly is illustrated in Fig. 3 as the boldly-drawn pattern. This trellis butterfly consists of a pair of input states (000, 001) and a corresponding pair of output states (000 and 100). For example, the second butterfly includes the input states 010 and 011 and the output states 001 and 101. Thus, the trellis diagram shown in the lower portion of Fig. 3 can be sub-divided into four trellis butterflies, i.e., the trellis butterflies I, II,

III and IV.

The decoding of a convolutional code is equivalent to the problem of reconstructing the most probable state sequence from the observed received code symbol sequence. In other words, the decoding task consists in finding the most probable path through the trellis diagram. In the art, this is also called Maximum Likelihood Sequence Estimation.

The Viterbi algorithm performs the maximum likelihood sequence estimation in an asymptotically optimum sense. The core of the Viterbi algorithm is the so-called Add-Compare-Select (ACS) recursion. For each state of the current time instant, the path metrics of the possible preceding states and the according transition metrics of the state transitions are added. Then, the path metrics from each branch leading to a state are compared to find out the most probable transition, i.e., the path having the highest or lowest path metric. Whether the highest or lowest path metric is used for comparing and selecting depends on the actual implementation. The state transition leading to, for example, the minimum path metric is then selected as the "surviving" path, which is also called the survivor.

In the following, a short example of this procedure is given in order to show how the survivor leading to the state 000 in cycle i+1 is obtained.

Firstly, the path metric of the path leading to the state 000 at the time instant i is retrieved and added to the transition metric for a transition from state 000 to state 000. This add-step yields a first candidate path metric. The path metric leading to the state 001 at time instant i is then retrieved and added to the transition metric for transition from the state 001 to 000 (time instant i+1). This second add operation will result in a second candidate path metric for the state 000 at time instant i+1. The first and the second path metrics with respect to the state 000 at

time instant i+1 are then compared to find out the most probable survivor path for this state. Depending on the metric definition, the more probable path is related to the minimum or maximum metric between these two candidate values, respectively. Accordingly, the path leading to the state 000 (time instant i+1) having the lowest or highest path metric is then selected as the surviving path, whereas the other path is discarded. This Add-Compare-Select operation is repeated for all remaining states, i.e., for the states 001 to 111 until all survivors for the time instant i+1, i.e., for the next cycle, are determined. When the next information symbol is received at the decoder, this procedure is repeated in order to determine the survivor path for the next cycle, the cycle i+2.

From this algorithm, a general structure of a Viterbi decoder, which is shown in Fig. 2, can be derived. The Viterbi decoder comprises an input 20 at which symbols are received that have been transmitted over a transmission channel, which can be, for example, a wire channel or a wireless channel. The received symbols are fed into a Transition Metric Calculation Unit (TMU) 22 which computes the transition metrics based on the received code symbols and the ideal symbols from the trellis diagram for each possible state transition. The Transition Metric Calculation Unit 22 is connected to an Add-Compare-Select-Unit (ACSU) 24 which is the core of the Viterbi decoder. The Add-Compare-Select-Unit 24 is connected to a path metric memory (PMM) 26, in which the path metrics of the surviving paths are stored. The ACSU 24 and the PMM 26 co-operate for performing the ACS recursion. The ACS recursion results in the ACS decisions, i.e., in a determination of the survivor paths, which are forwarded to a Survivor Memory Unit (SMU) 28 which stores all surviving paths and determines the finally decoded information bit stream which is available at an output 30 of the Viterbi decoder. As it is known in the art, several possibilities exist for outputting the decoded information bits. Depending on the length of the coded

signal, which is the signal received by the Viterbi decoder, the survivor path sub-word in the survivor memory SMU 28 would grow indefinitely long as time proceeds. To avoid this, the survivor path sub-word must be periodically truncated and the oldest symbol is discarded. Those discarded symbols have a high tendency to agree and to match the true message, they are the output of the Viterbi decoder.

In the following, the Memory Management of the path metric memory will be discussed. Computing the path metrics in the Add-Compare-Select-Unit 24 has to be performed with respect to the causality of the operations, i.e., the memory contents of the old path metrics of the preceding time instant (cycle) must not be overwritten, unless the corresponding path metric is not required for the computation of the new path metrics anymore. It would be natural to treat the contents of the shift register as a k-digit M-ary number and use this number to address the path metric memory. However, such an addressing scheme is inconsistent with writing new metrics over old metrics. When Fig. 3 is considered, it can be seen that the butterfly I includes the output states 000 and 100. When the path metric of each state is stored in the path metric memory under an address which corresponds to the number of the states, the path metric for the state 000 would be stored under the address 000. Accordingly, the path metric of the state 001 would be stored under the address 001, and so on. Since the first butterfly results in the state 000, the path metric for the state 000 and the time instant or cycle i+1 could be stored at the address 000, since the path metric for the state 000 at the time instant i is not required anymore. However, this procedure cannot be conducted for the state 100, since the path metric of the state 100 for the time instant i must not be overwritten. The path metric for the time instant i is still required to compute the butterfly III. Thus, it would be necessary to double buffer the path metric memory. In other words, such an addressing

organisation would need two path metric memories on the first sight, which are alternately read or written ("ping-pong memories").

A memory management which overcomes this disadvantageous double-buffering scheme is described in "Memory Management in a Viterbi Decoder", C.M. Rader, IEEE Transactions on Communications, Vol. COM-29, No. 9, pages 1399 to 1401, September 1981. The memory management described in this reference is called an "In-place" Memory Management. The addressing scheme of such an in-place memory management will be described with reference to Fig. 9. It is to be noted that Fig. 9 is related to the trellis diagram shown in Fig. 3. Fig. 9 shows a path metric memory having 8 memory cells that can be individually addressed by the addresses 000 to 111 which are given in the column at the left of Fig. 9. Each memory cell comprises a length such that the path metric of the survivor path ending in the respective states can be stored. It is assumed that the memory addresses correspond to the state numbers in the cycle 0. In other words, the memory addresses and the state numbers are both in their natural order, in which a lower number precedes a higher number.

When a new information bit is received, the survivor path for cycle 1 has to be computed. The state transitions from cycle 0 to cycle 1 have to be calculated using the transition metrics that depend on the newly-received information symbol. It is to be noted with respect to Fig. 9 that the circles 90 between the cycles represent an ACS device which is arranged for processing one butterfly. The lines leading from the cycle columns to the circles represent the inputs and outputs of the ACS device with respect to the location in the path metric memory.

The addressing scheme described by Rader will be summarized below. An ACS device 90 retrieves path metrics for the state 000 and 001 and outputs path metrics for the states 000 and

100. In contrast to the simple memory management system described above, the path metric for the state 100 is stored at the address 001, rather than at the address 100. This memory management is called an in-place management, since the results of one ACS device are stored at the same location from which the inputs for this ACS device have been retrieved. In another sub-cycle, the ACS device receives, as inputs, the path metrics for the states 010 and 011. It outputs path metrics for the states 001 and 101. This ACS device is then loaded with path metrics for the states 100 and 101 in order to output path metrics for the states 010 and 110. Finally, in a fourth sub-cycle, the ACS device is fed with path metrics for the states 110 and 111 in order to output survivor path metrics for the states 011 and 111.

After the fourth sub-cycle, the first cycle is finished. In the path metric memory, a full set of survivor path metrics is stored now. Because of the in-place memory management, the path metrics, however, are not stored in their natural order, but in the order shown in the column, cycle_1, of Fig. 9. When a new information bit is received, the survivor path metrics for cycle_2 have to be calculated. In this cycle_2, the addressing scheme, however, has to be changed with respect to the calculating of cycle_1. In particular, the path metric memory has to be processed such that the ACS device 90 is fed with the path metric for the state 000 at memory address 000 and with the path metric of the state 001 at memory address 010. This addressing scheme is continued for four sub-cycles until cycle_2 is completed. Analogously, this procedure is repeated for the computation of cycle_3.

It can be seen that computing metrics in-place leads, inevitably, to an addressing scheme which changes after each decoding cycle. In general, the metrics accessed together for one ACS device are found by generating their natural addresses, but rotating the bits of these addresses by i places to the left before reading (or writing) the metrics from (or into) the memory. The natural addresses can be

generated from a counter, for example.

This memory management method is also memory efficient in that no double buffering of the path metric memory is needed. However, it is only applicable for a serial computation of the Viterbi algorithm, i.e., only one trellis butterfly can be computed in one time instant. Since this in-place memory management method is only suited for fully serial Viterbi decoders, no scalable Viterbi decoders that combine serial processing with parallel processing have been built around this known in-place memory management method.

In the prior art, several parameterizable Viterbi decoder architectures are known. Some of them are described in the following references:

S. Bitterlich, B. Pape, H. Meyr, "Area Efficient Viterbi-Decoder Macros", Proceedings of the 1994 European Solid-State Circuits Conference (ESSCIRC'94), Ulm, Germany;

H. Dawid, S. Bitterlich, H. Meyr, "Trellis Pipeline-Interleaving: A Novel Method for Efficient Viterbi-Decoder Implementation", Proceedings IEEE ISCAS (San Diego), pages 1875-1878, 1992;

G. Feygin, P.G. Gulak, P. Chow, "Generalized Cascade Viterbi Decoder - A Locally Connected Multiprocessor with Linear Speed-Up", ICASSP, pages 1097-1100, 1991; and

P.G. Gulak, Th. Kailath, "Locally Connected VLSI Architectures for the Viterbi Algorithm", IEEE Journal on Selected Areas in Communications, Vol. 6, No. 3, April 1988.

For all these architectures, a complicated control is required due to the complex data flow and the complex computation mechanisms. Furthermore, storage of the path or state metrics of the Viterbi algorithm as well as the implementation of data flow paths is performed using

registers which are inefficient considering area and power consumption when compared to common Random Access Memories (RAM).

The object of the present invention is to provide a scalable Viterbi decoder that can be implemented in an easy and efficient way.

This object is met by a Viterbi decoder in accordance with claim 1.

The present invention is based on the finding that all trellis butterflies which are computed in one processing cycle can be grouped to so-called super butterflies. This allows the interpretation of the ACS computation with accordingly fewer equivalent states. By providing an ACS unit having at least two ACS devices followed by a preferably hard-wired output stage, the path metric memory addressing scheme can be considerably simplified. The number of memory cells, which have to be addressed, corresponds to the equivalent states, rather than to the "natural" states. When, for example, a trellis diagram having eight states (four butterflies) is considered, only four equivalent states have to be addressed when two ACS devices are connected in parallel. Each memory cell of the four memory cells for the four equivalent states in this example has a length that equals the length necessary for storing two path metrics. In general, the length of a memory cell adapted for storing one equivalent state corresponds to the place necessary for storing a number of path metrics, the number corresponding to the number of ACS devices connected in parallel. The scalability is reached by the processing of one cycle in a plurality of sub-cycles, such that for each specific application, an optimal tradeoff between speed and power or area consumption can be obtained.

The Viterbi decoder in accordance with the present invention

is based on an in-place memory management method, that is memory efficient in that no double buffering of the path metric memory is required. Furthermore, the memory management method of the inventive Viterbi decoder does not require the use of registers or complicated addressing schemes, but is suitable for power efficient and readily available one-port RAMs.

In contrast to the in-place memory management disclosed by Rader, which is only suited for a fully serial Viterbi architecture, the memory management method in accordance with the present invention allows the implementation of scalable Viterbi decoders, which require only a left rotation of the bits of the addresses for the equivalent states, rather than of the bits of the natural addresses. Thus, less bits have to be left rotated than in the fully serial architecture. The grouping of the output states of the several parallel ACS devices is performed by means of a simple grouping instruction which requires that each memory cell receives one path metric from each ACS device per sub-cycle and which requires that one memory cell includes the path metrics required for the processing of one trellis butterfly. This combining instruction is readily suitable for a hard-wired implementation which is preferable both for cost and performance reasons.

In the following, preferred embodiments of the present invention are described in detail with reference to the accompanying drawings, in which:

Fig. 1 shows a general structure for a convolutional encoder;

Fig. 2 is a straightforward block diagram of a Viterbi decoder;

Fig. 3 illustrates a general convolutional encoder and the corresponding trellis diagram for a convolutional

coder having a constraint length of 4 and a code rate of 1/2;

Fig. 4 illustrates a scalable in-place memory management method in accordance with the present invention for a trellis diagram having eight natural states and four equivalent states;

Fig. 5 shows a block diagram of a part of an inventive Viterbi decoder adapted for implemention of the memory management illustrated in Fig. 4;

Fig. 6 shows a more detailed view of an inventive Viterbi decoder;

Fig. 7 shows a memory management method for a scalable Viterbi decoder having four parallel ACS devices and a trellis diagram having sixteen natural states corresponding four equivalent states;

Fig. 8 is a general block diagram of an inventive Viterbi decoder having an arbitrarily selected number of parallel ACS devices; and

Fig. 9 shows a prior art in-place memory management method for a fully serial Viterbi decoder architecture.

With reference to Fig. 4 and Fig. 5, a Viterbi decoder in accordance with the present invention will be described. For the sake of clarity, the teachings of the present invention will be set forth by means of the trellis diagram shown in Fig. 3, which has only eight states.

When an inventive Viterbi decoder is used for decoding a signal which has been encoded by a convolutional code which can be represented by the trellis diagram shown in Fig. 3, a Viterbi decoder in accordance with the present invention will comprise only two ACS devices. Each ACS device will

process one trellis butterfly in the first sub-cycle and another trellis butterfly in another sub-cycle such that all four butterflies will be processed in one cycle.

The term "cycle" covers the full Viterbi decoder processing done for one received symbol. Since the inventive Viterbi decoder is a scalable decoder, each cycle comprises at least two sub-cycles.

The term "sub-cycle" is used for describing the operating task conducted by one ACS device. The operating task of one ACS device in turn includes loading the ACS device, performing the add operation, performing the compare operation, and performing the select operation and, finally, the outputting of the new path metrics for the surviving paths which are also called survivors.

The processing task of one ACS device can be sub-divided into individual steps. The term "step", thus, relates to the individual processing actions of one ACS device. One step may coincide with one clock period. Based on this terminology, one sub-cycle comprises a first address step for loading old path metrics into the ACS device. Usually, the loading of the old path metrics into an ACS device will not take a full cycle. When sufficiently fast chips are used for an ACS device, the add-compare-select operation will be finished in the same clock cycle, i.e. the step in which the old path metrics were loaded into the ACS device. Thus, the second step again is an addressing step in which the new path metrics are output from the ACS device and stored in the path metrics memory.

The grouping of all in one read-write cycle computed trellis butterflies to so-called super butterflies allows the interpretation of the ACS computation with accordingly fewer equivalent states. The memory scheme given in Fig. 4 comprises eight states, but only four equivalent states which can be addressed by only two bits as it is shown in

Fig. 4 by the column titled memory address.

The states are grouped according to identical most significant bits (MSB) of their ν binary state address bits. The states are grouped into sets where only the μ least significant bits (LSB) are differring. Thus, the ν minus μ MSBs are identical and determine the number of the equivalent state. Accordingly, the number of equivalent states is given by the following equation.

$$S_{eq} = S/C = 2^{\nu}/2^{\mu} = 2^{(\nu-\mu)} \qquad \text{(Equation 3)}$$

In Equation 3, S is the number of states, and C is the number of ACS devices. The grouping into super states allows simplified addressing of the path metrics memory by an in-place algorithm which has only to account for the MSBs of the state addresses, i.e. only ν minus μ MSBs have to be left-rotated instead of the ν required in the Rader algorithm which has been described with reference to Fig. 9. In the Rader algorithm, for the same trellis diagram, three bits have to be left rotated. In constrast thereto, the inventive Viterbi decoder only involves the left-rotating of two bits allowing a simplified address generator.

Accordingly, $C = 2^{\mu}$ butterfly ACS computers are required to perform the ACS operations for the 2C corresponding pairs of input and output states in parellel. In contrast thereto, a serial Viterbi decoder utilizing the Rader algorithm is able to compute only two states in parallel ($\mu = 0$). The advantage of the new algorithm is that, with increasing parellelism, i.e. with increasing μ, accordingly fewer equivalent states have to be addressed in the path metrics memory, whereby the memory management can be simplified.

Fig. 4 shows the inventive in-place memory management method for a Viterbi decoder with eight states in two parallel ACS computers ($\nu = 3$, $S = 8$, $\mu = 1$, $C = 2$). The ACS operations are computed for $2C = 4$ states in parallel. The equivalent

state number is $S_{eq} = 8/2 = 4$. Therefore, only four instead of eight memory cells have to be addressed in the path metrics memory. As can be seen from Fig. 4, only the $\nu - \mu = 3 - 1 = 2$ MSBs have to be left-rotated for each decoding of a trellis state, i.e. a cycle. Thus, the metrics of all states with the state number of $S = (0\ 1\ s_0)$ which corresponds to the equivalent state address $A = (0\ 1)$ are stored under the memory address $A = (1\ 0)$ in the next cycle. Thus, only a very simple control mechanism is required. The memory words consist of the path metrics of all states grouped to a super state in correct order. The grouping is done by an output stage which combines the survivor path metrics output by the plurality of ACS devices such that each memory still receives one path metrics from each ACS device per sub-cycle and that one memory cell includes the path metrics required for the processing of one trellis butterfly.

In the following, the detailed processing of the memory scheme given in Fig. 4 by means of the Viterbi decoder shown in Fig. 5 will be described.

Fig. 5 shows a certain portion of a Viterbi decoder which has the general structure which has already been described with reference to Fig. 2. The ACS unit 24 includes the first ACS device ACS_0 50 and a second ACS device ACS_1 52 which are connected in parallel. The ACS unit 24 further comprises an output stage 54 for performing the grouping of the path metrics output by the ACS devices 50, 52 in order to construct a first memory word 56 and a second memory word 58. The ACS device 50 comprises two inputs 60a and 60b and two outputs 60c and 60d. Similarly, the second ACS device ACS_1 52 includes two inputs 62a and 62b as well as two outputs 62c and 62d. The inputs of the ACS devices 50 and 52 are connected to a memory interface 64. The output stage 54 is connected to another memory interface 66. The memory interfaces 64 and 66 can be implemented, for example, by multiplexers or other devices known in the art.

Before the memory addressing scheme given in Fig. 4 is described in detail, short reference is made to Fig. 6 which shows the full range of the ACS unit and path metrics memory cooperation. The path metrics memory 26 and the ACS unit 24 are connected by the memory input interface 64 and the memory output interface 66. Additionally, both elements are connected to a scalable in-place memory management control device 68 which includes, as a key element, an address generator for generating addresses for addressing the path metrics memory in order to load the ACS units with old path metrics and store new path metrics from the ACS devices into the path metrics memory.

It can be seen that Fig. 5 and 6 already correspond to each other. However, Fig. 5 shows a somewhat more detailed view of the memory input interface 64 and the memory output interface 66 as well as the output stage 54. The memory input interface 64 in Fig. 5 comprises two switches 64a and 64b which are only used for illustrating that during the time, in which ACS_0 is connected to the memory, ACS_1 is disconnected from the memory. This function can be implemented by any element known in the art which can behave accordingly. A preferred implementation will be based on multiplexers.

Referring to Fig. 5, the output stage 54 consisting of a hard wired implementation is shown, in which the first output 60c of the first ACS device 50 and the first output 62c of the second ACS device 52 are connected to output or group the first memory word, whereas the respective second outputs are connected to group the second memory word. For speed reasons and for the sake of chip simplicity, it is preferred that the output stage is hard wired. The hard wired scheme for the output stage 54 is possible, since the grouping remains the same over each cycle. Alternatively, this grouping, however, of the new path metrics output by the ACS devices can also be implemented using electronic

multiplexers. Finally, the output memory interface 66 is also schematically shown as including a switch 60 therein which is intended to symbolize that either the first memory word 56 or the second memory 58 is stored in the path metrics memory. Of course, the switch 67 can also be implemented as an electronic multiplexer.

It has to be noted here that the input memory interface 64 and the output memory interface 66 are designed for connecting the ACS unit 24 with a single port RAM memory as the path metrics memory. In a single port RAM, only one read or write operation can be performed in one addressing step. Since they are cheap and readily available, single port RAM memories are preferred for constructing the path metrics memory of the inventive Viterbi decoder. However, the memory input interface and the memory output interface can also be designed for dual port RAMs in which a write step as well as a read step can be performed simultaneously.

In the following, a full description of the ACS unit addressing in subsequent cycles will be given. It is presumed that the path metrics memory has stored path metrics for the states in the natural order. This means that, at the memory address 00, the path metrics for the states 000 and 001 are stored. Accordingly, at memory address 01, the path metrics for the states 010 and 011 are stored. Analogously, the path metrics for the shift register states 100 and 101 are stored at the memory address 10. Finally, the path metrics for the register states 110 and 111 are stored at memory address 11.

In a first addressing step, the address generator 68 (Fig. 6) generates the read address 00 resulting in loading the path metrics for the states 000 and 001 into ACS_0. In a second addressing step, the address generator addresses memory address 01 in order to load the path metrics for the states 010 and 011 into ACS_1. As it can be seen from the trellis diagram in Fig. 3, ACS_0 calculates a new path

metrics for the register states 000 and 100. Furthermore, ACS_1 generates the new path metrics for the states 001 and 101.

If the output stage 54 were not present, the first memory word would consist of path metrics for the states 000 and 100. However, these states are located in different trellis butterflies. Therefore, the inventive Viterbi decoder includes the output stage which combines the respective first outputs 60c and 62c of the ACS devices such that the first memory word again consists of the path metrics for the states 000 and 001, which states being located in the same trellis butterfly, when Fig. 3 is considered. The output stage further combines the respective second outputs 60d and 62d of the ACS devices such that the path metrics for states 100 and 101 are grouped to form the second memory word 58 which comprises the path metrics for trellis butterfly III (Fig. 3).

Since the inventive Viterbi decoder is based on an in-place memory management, the first memory word is stored at address 00, i.e. at the same address from which the input of ACS_0 has been read. Additionally, the second memory word 58 is stored at memory address 01, i.e. the memory address from which the inputs for ACS_1 have been retrieved.

It has become clear from the above that a first sub-cycle is finished now. The first sub-cycle thus comprises one addressing step for loading ACS_0, one addressing step for loading ACS_1, one addressing step for storing the first memory word, and one addressing step for storing the second memory word. Since the memory management method is an in-place memory management method, the first memory word is stored at the same address, from which ACS_0 has been loaded, and the second memory word is stored at the same address from which ACS_1 has been loaded. In the second sub-cycle, the address generator generates memory address 10 for loading ACS_0 with path metrics for the states 100 and 101.

Then, the address generator generates the memory address 11 for loading ACS_1 with path metrics for the states 110 and 111. ACS_0 generates the path metrics for the survivor states 010 and 110 and ACS_1 generates the path metrics for the survivor states 001 and 111. The output stage 54 performs grouping of the respective outputs such that the first memory word now consists of the path metrics for the states 010 and 011, whereas the second memory word now consists of the path metrics for the states 110 and 111.

After two addressing steps for storing the first memory word and the second memory word in the path metrics memory, the second sub-cycle is finished and also the first full cycle is completed. Now, the path metrics memory addressing scheme is the scheme titled cycle_1 in Fig. 4. It can be seen that the states stored in memory addresses 01 and 10 have their two most significant bits left rotated in comparison to the preceding cycle_0.

When the Viterbi decoder receives the new information symbol, another cycle is started. In a first addressing state of the second sub-cycle, the address generator generates a memory address 10 such that ACS_0 receives the path metrics for states 100 and 101. Then, the address generator generates memory address 11 resulting in ACS_1 receiving the path metrics for the states 110 and 111. The first memory word consists of the new path metrics for the states 010 and 011, whereas the second memory word consists of the new path metrics for the states 110 and 111. The first memory word is stored at memory address 01 and the second memory word is stored at memory address 11. Now, cycle_1 of Fig. 4 is finished.

When a new input symbol is received, another cycle is started. The processing will be identical to the processing described above for the first cycle. However, the address generator will work slightly differently. In the first sub-cycle, the address generator will generate the memory

address 00 for loading ACS_0 and memory address 10 for loading ACS_1. In the second sub-cycle, the address generator will generate address 01 for loading ACS_0 and address 11 for loading ACS_1. Thus, it can be seen that the address generator has left-rotated the bits of all read addresses. It is pointed out that the bitwise left-rotation of addresses 00 and 11 does not change the respective addresses. However, a left-rotation of bits will indeed change addresses 01 and 10 to 10 and 01, respectively.

In the following, reference is made to Fig. 6. Fig. 6 shows in detail which path metrics are loaded into the ACS devices during the reading steps and the writing steps. In the first step, the path metrics called $GAMMA_{old0}$ and $GAMMA_{old1}$ are loaded into ACS_0, and in the second step, $GAMMA_{old2}$ and $GAMMA_{old3}$ are loaded into ACS_1. In the first step $GAMMA_{new0}$ and $GAMMA_{new1}$ are loaded into the path metrics memory as the first memory word, and in the second step $GAMMA_{new2}$ and $GAMMA_{new3}$ are loaded into the path metrics memory as the second memory word. In the second sub-cycle, the number of 4 (four) has to be added to each index for obtaining the correct path metrics for a second sub-cycle. For example, $GAMMA_{old4}$ and $GAMMA_{old5}$ are loaded into ACS_0 and $GAMMA_{new4}$ and $GAMMA_{new5}$ are output from ACS_0 and written into the path metrics memory.

Although it has not been described in detail before, the ACS devices 50 and 52 also need their respective transition metrics from the transmission metric unit 22 (Fig. 2). Additionally, the ACS decisions from the ACS units have to be transmitted to the survivor memory 28 (Fig. 2).

To summarize, Fig. 6 shows an example Viterbi decoder architecture having two parellely arranged ACS computers, each ACS computer performing two serial sub-cycles. The old path metrics $GAMMA_{old}$ are read from the path metrics memory in two subcycles for two super states, wherein each super state comprises $2^{\mu} = 2$ basic states. The ACS unit computes

new path metrics $GAMMA_{new}$ from the old path metrics and the corresponding branch metrics LAMBDA multiplexed from the transition metrics unit for 2C = 4 states in parallel. The resulting new path metrics are grouped to the according two new super states by the preferably fixed wiring scheme of the ACU output stage. Then, the new path metrics $GAMMA_{new}$ are stored in-place, according to the new in-place memory management method, in the same memory cells from which the according old path metrics have been read.

The selected positions survivor paths are passed to the survivor memory unit of the Viterbi decoder for final decoding of the received information symbols. The entire processing is controlled by the scalable in-place memory management control device which performs the generation of addresses and control signals for the path metrics memory. The number of PMM cells corresponds to the number S_{eq} = 4 of the equivalent states. Each memory cell consists of the path metrics of C = 2 basic states, i.e. the word length of the memory is C x PML = 2 x PML, wherein PML is the word length of the path metrics.

Fig. 8 shows a generalized view of the inventive Viterbi decoder having an arbitrarily selected number of ACS devices. To illustrate the versatility of the scalable Viterbi decoder architecture in accordance with the present invention, Fig. 7 shows an exemplary scalable in-place memory management scheme for a trellis diagram having sixteen states, the sixteen states being processed by four parallel ACS devices, such that four equivalent states S_{eq} exist. In contrast to the first example given in Fig. 4 in which the length of one memory cell corresponds to the length of two path metrics, the length of one memory cell for the example illustrated in Fig. 7 corresponds to the length of four memory words. In the first sub-cycle of cycle_0, the address generator 68 generates 0 for loading ACS_0 and ACS_1. Then, in the second addressing step of the first sub-cycle, the address generator 68 generates address

01 for loading ACS_2 and ACS_3. Now, the output stage is implemented such that the first memory word is formed by the first outputs of all four ACS devices and the second memory word is formed by the second outputs of all four ACS devices.

After storing the first and the second memory words in-place in the path metrics memory, the address generator generates address 10 for loading ACS_0 and ACS_1 and memory address 11 for loading ACS_2 and ACS_3. Again, the first memory word consists of the first outputs of all four ACS devices and the second memory word consists of the second outputs of all ACS devices. This procedure is repeated for a newly received symbol with the exception that the address generator left-rotates the address bits such that the memory address for loading ACS_2 and ACS_3 in the first cycle is "10" instead of "01" as before and that the memory address for loading ACS_0 and ACS_1 in the second sub-cycle is "10" instead of "01" as before.

It is to be noted that circles between the columns in Fig. 7, Fig. 4 as well as in Fig. 9, give an indication of the processing in one sub-cycle. Therefore, the number of circles between the columns in the respective figures corresponds to the number of sub-cycles in one cycle. Additionally, the lines from the circles to the columns indicate the respective inputs and outputs of the respective ACS unit in one sub-cycle.

Although the memory management method given in Fig. 7 is implemented by four ACS devices, the trellis diagram having sixteen states can also be processed by two ACS devices. Then μ equals 1, such that three instead of two bits are left-rotated for achieving an in-place memory management. In the case of two ACS units for sixteen basic states, four sub-cycles instead of two sub-cycles are required, and the bit rotation corresponds to the bit rotation shown in Fig. 9. However, the number of basic states is sixteen instead of

eight.

The inventive Viterbi decoder, thus, provides a scalable solution which is flexible for adapting to any specific application. When a designer wishes to select the number of parallely arranged ACS devices, he will check which data rate is required for his specific application. Then, he will determine the number of sub-cycles, i.e. number of ACS devices, based on his knowledge of the processing speed of a single ACS device. This procedure will give an optimal solution for the tradeoff between chip area and power consumption on the one hand and processing speed on the other hand.

Claims

1. Viterbi decoder for decoding a signal represented by a sequence of symbols, the signal being encoded by means of a convolutional code representable by at least four independent trellis butterflies, each trellis butterfly having at least two input states and at least two output states, comprising:

a transition metric calculation unit (22) for calculating, in one cycle, transition metrics from current states to following states based on a symbol received in a cycle;

an add-compare-select unit (24) for calculating surviving parts and path metrics of the surviving paths, based on the current states and the transition metrics calculated by the transition metrics calculation unit (22);

a survivor memory unit (28) for storing the surviving paths calculated by the add-compare-select unit (24) for providing a decoded signal;

a path metrics memory (26) for storing the path metrics of the surviving paths calculated by the add-compare-select unit (24); and

a path metrics memory address generator (68) for generating read addresses for retrieving path metrics input into the add-compare-select unit (24) and for generating write addresses for storing the path metrics calculated by the add-compare-select unit (24),

wherein the add-compare-select unit (24) comprises a number of parallely arranged add-compare-select devices (50, 52), the number of add-compare-select devices being greater than or equal two and smaller than the number of

trellis butterflies, each add-compare-select device (50, 52) being arranged for processing at least two trellis butterflies in at least two subsequent sub-cycles, wherein the cycle comprises at least two sub-cycles,

wherein the path metrics memory (26) comprises individually addressable memory cells, the size of each cell being such that a number of path metrics can be stored in one cell, the number of path metrics stored in one cell being equal to the number add-compare-select devices,

wherein the path metrics memory address generator (68) is arranged for generating the write addresses which are identical to the read addresses, and for generating read addresses for a subsequent cycle by left-rotating the bits of the read addresses of a preceding cycle, and

wherein the add-compare-select unit further comprises an output stage (54) for combining survivor path metrics output by the plurality of add-compare-select devices such that each memory cell receives one path metric from each add-compare-select device per sub-cycle, and that one memory cell receives the path metrics required for the processing of one trellis butterfly.

2. Viterbi decoder in accordance with claim 1, wherein the path metrics memory (26) is implemented as a random access memory, and preferably as a single port random access memory.

3. Viterbi decoder in accordance with claim 1 or claim 2, wherein the signal encoded by means of the convolutional code is a K-ary signal, K being an integer, wherein the convolutional code can be represented by a shift register having L stages, the stages including one input stage and L - 1 memory stages (10b,10c), the shift register having S states wherein $S = 2^{k(L-1)}$ and B state

transitions, wherein $B = 2^{kL}$, wherein the number of add-compare-select devices (50, 52) being defined as $k(L-1)-\mu$, μ being an integer greater than or equal to 1 and smaller than $k(L-1)$,

wherein the size of one individually addressable path metrics memory cell being 2μ times the number of bits required for storing a path metric of a single survivor path,

wherein the path metrics memory addresses correspond to the $[k(L-1)-\mu]$ most significant bits of the S state addresses, and

wherein the path metrics memory address generator (68) is arranged for left-rotating the $[k(L-1)-\mu]$ most significant bits of each of the S states to generate the corresponding write address.

4. Viterbi decoder in accordance with any one of the preceding claims, wherein the individually addressable memory cells are arranged for sequentially storing the path metrics obtained by processing one trellis butterfly.

5. Viterbi decoder in accordance with any one of the preceding claims, wherein the output stage (54) of the add-compare-select unit (24) comprises a fixed wiring scheme which is arranged for combining the respective first output states (60c, 62c) of the plurality of add-compare-select devices (50, 52), and for combining the second output states (60d, 62d) of the plurality of add-compare-select devices (50, 52) in first memory word (56) and a second memory word (58), wherein the first memory word (56) is stored in the path metrics memory (26) at an address from which the inputs (60a, 60b) for the first add-compare-select device (50) have been retrieved, and wherein the second memory word (58) is

stored in the path metrics memory (26) at an address from which the inputs (62a, 62b) for the second add-compare-select device (52) have been retrieved.

6. Viterbi decoder according to any one of the preceding claims, wherein the memory cells in the path metrics memory (26) are arranged for sequentially storing the number of metrics in a natural order in which a higher state is stored after a lower state, and wherein the Viterbi decoder comprises a hard wired memory input interface (64) which is arranged for directing path metrics in the natural order to the first add-compare-select device (50), and if the memory cell includes more than two path metrics, to the second add-compare-select device.

7. Viterbi decoder in accordance with any one of the preceding claims, wherein the transition metrics are parallely multiplexed to the plurality of add-compare-select devices (50, 52).

8. Viterbi decoder according to any one of the preceding claims, wherein the trellis butterflies are ordered such that a trellis butterfly including lower input states is positioned before a trellis butterfly including higher input states, and

wherein the add-compare-select devices (50, 52) are arranged to process the trellis butterflies in accordance with their natural order in one sub-cycle, such that two neighbouring trellis butterflies are processed by different add-compare-select devices (50, 52) in one subcycle.

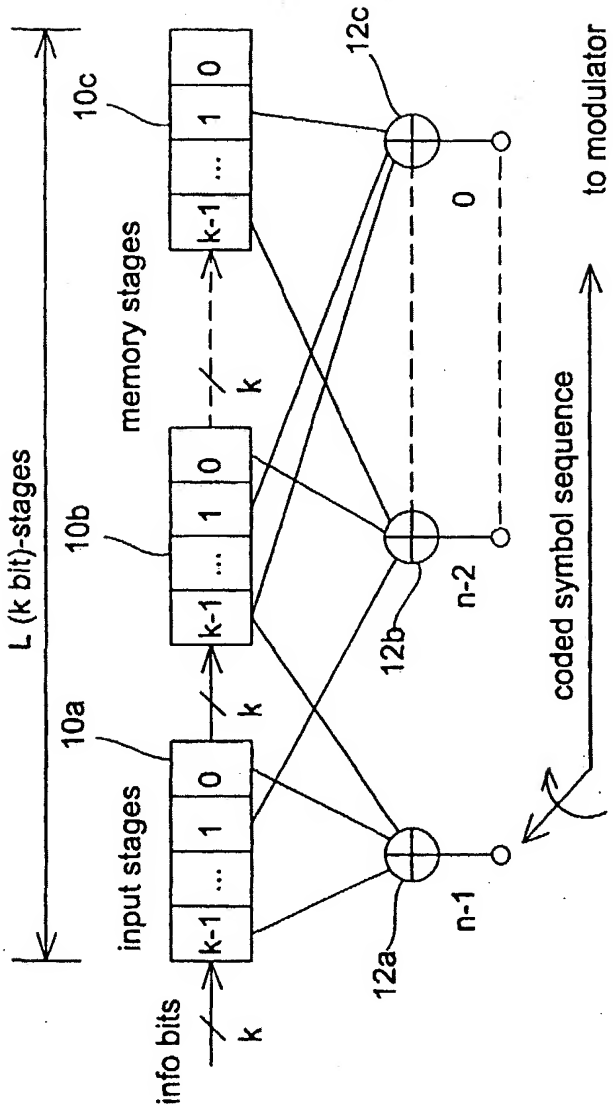

L (k bit)-stages
10a
10b
10c
memory stages
input stages
info bits
k
k-1
...
1
0
12a
12b
12c
n-1
n-2
0
coded symbol sequence
to modulator

Fig. 1

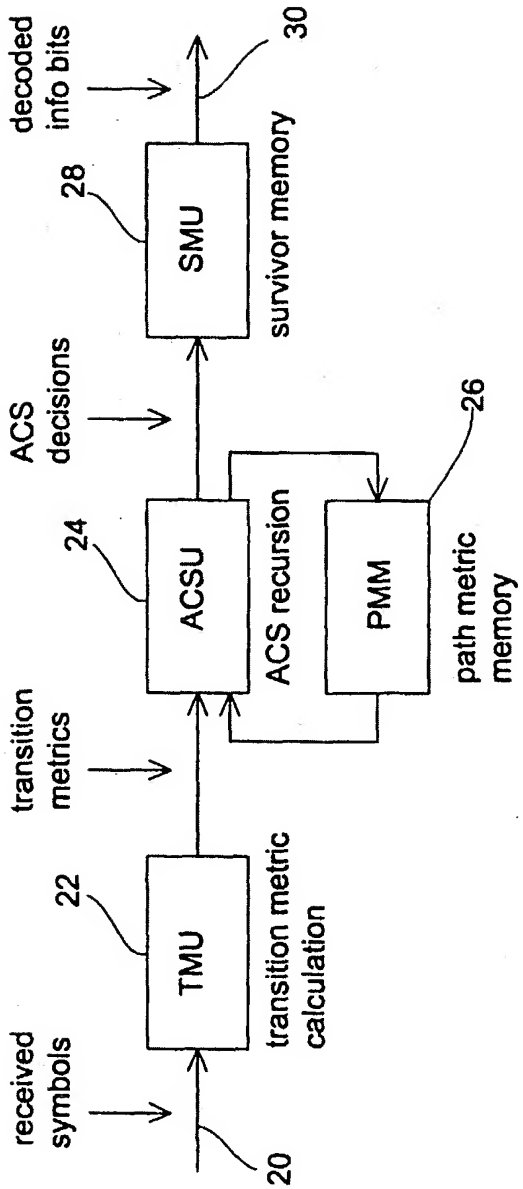

received symbols
20
TMU
22
transition metric calculation
transition metrics
ACSU
24
ACS recursion
PMM
26
path metric memory
ACS decisions
SMU
28
survivor memory
decoded info bits
30

Fig. 2

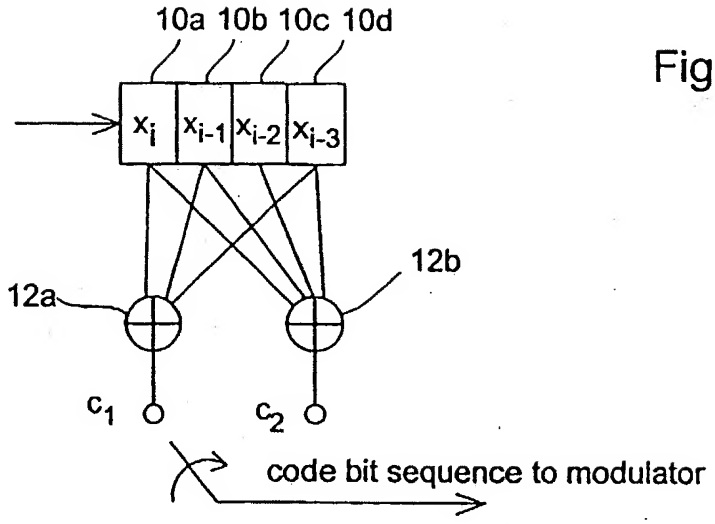
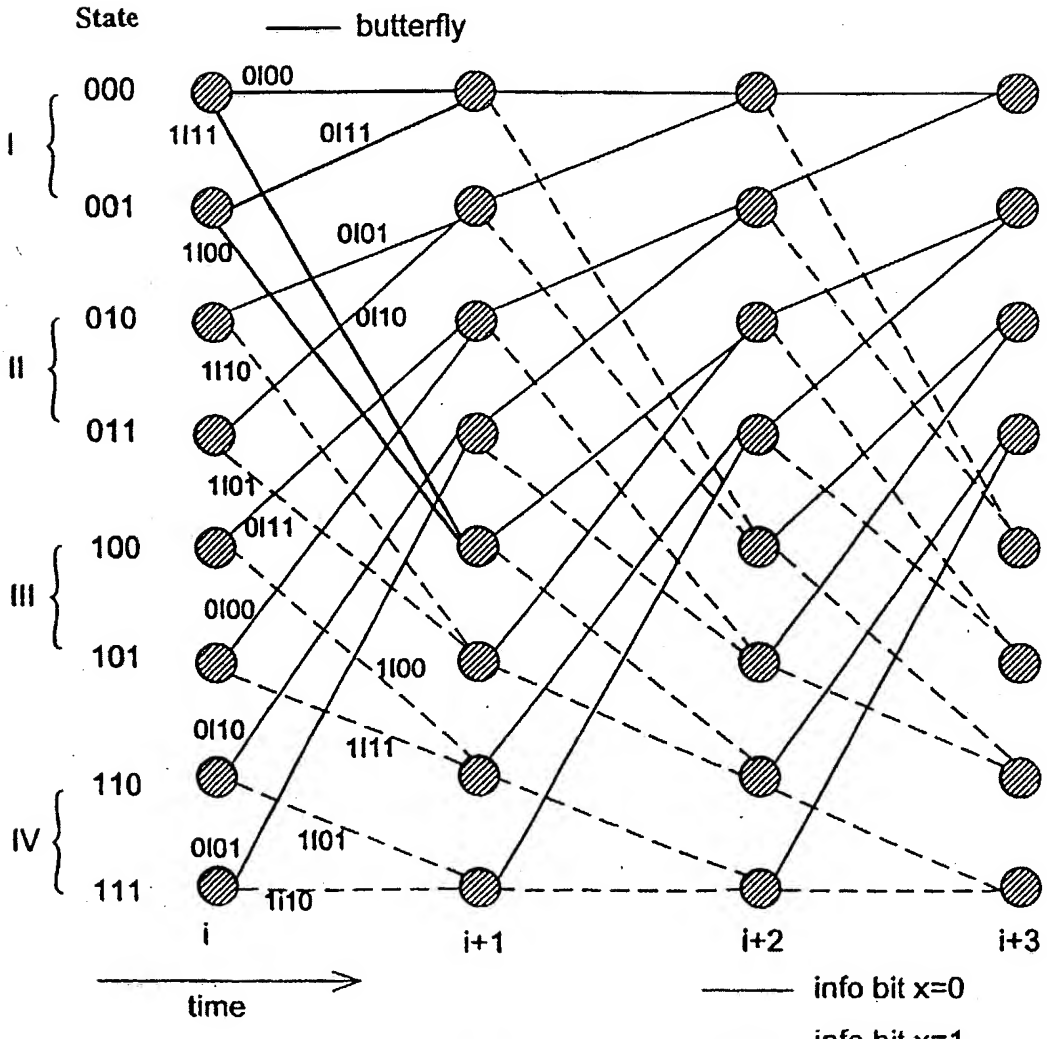

10a 10b 10c 10d
x_i x_{i-1} x_{i-2} x_{i-3}
12a
12b
c_1
c_2
code bit sequence to modulator

Fig. 3


State
butterfly
000
001
010
011
100
101
110
111
I
II
III
IV
0100
1111
0111
0101
1100
0110
1110
1101
0111
0100
1100
0110
1111
0101
1101
1110
i
i+1
i+2
i+3
time
info bit x=0
info bit x=1

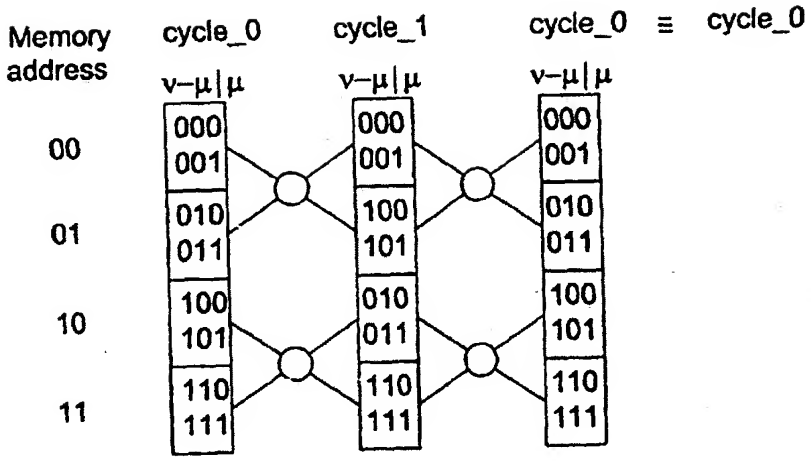

Memory address
cycle_0
cycle_1
cycle_0 ≡ cycle_0
ν–μ|μ
ν–μ|μ
ν–μ|μ
00
01
10
11
000
001
010
011
100
101
110
111
000
001
100
101
010
011
110
111
000
001
010
011
100
101
110
111

Fig. 4

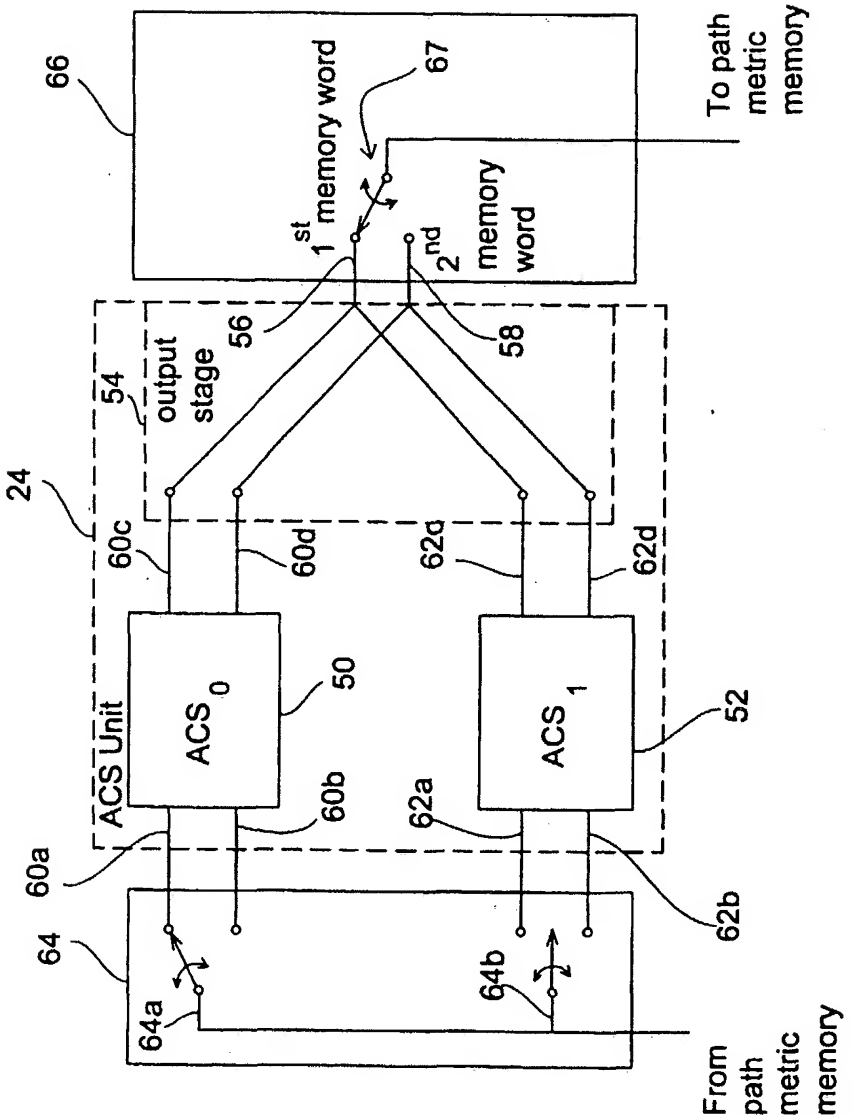

66
67
To path metric memory
1st memory word
2nd memory word
56
58
54
output stage
24
60c
60d
62c
62d
ACS Unit
ACS0
50
ACS1
52
60b
62a
60a
62b
64
64a
64b
From path metric memory

Fig. 5

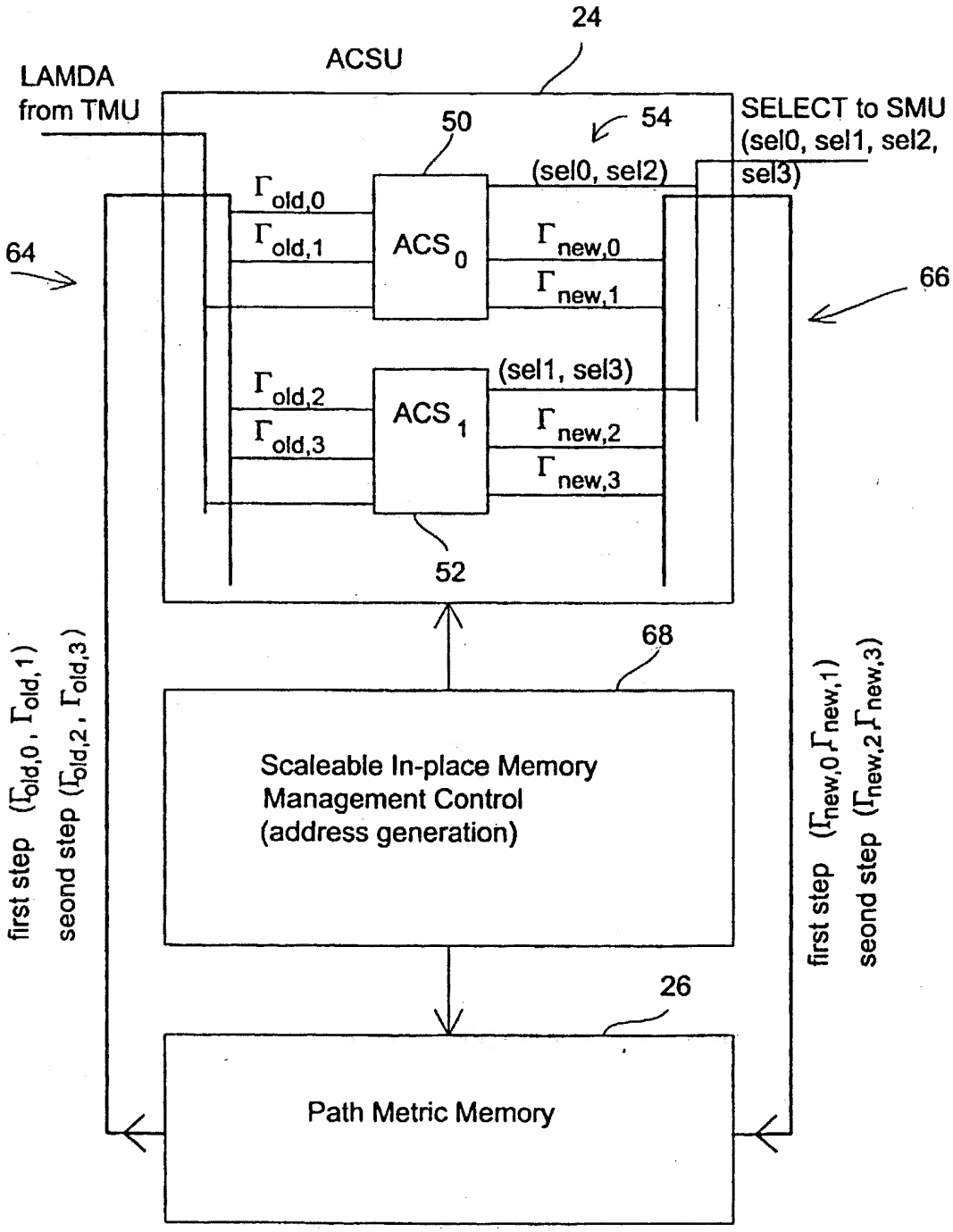

24
ACSU
LAMDA
from TMU
50
54
SELECT to SMU
(sel0, sel1, sel2,
sel3)
(sel0, sel2)
$\Gamma_{old,0}$
$\Gamma_{old,1}$
ACS_0
$\Gamma_{new,0}$
$\Gamma_{new,1}$
64
66
(sel1, sel3)
$\Gamma_{old,2}$
$\Gamma_{old,3}$
ACS_1
$\Gamma_{new,2}$
$\Gamma_{new,3}$
52
68
Scaleable In-place Memory
Management Control
(address generation)
first step ($\Gamma_{old,0}$, $\Gamma_{old,1}$)
seond step ($\Gamma_{old,2}$, $\Gamma_{old,3}$)
first step ($\Gamma_{new,0}$,$\Gamma_{new,1}$)
seond step ($\Gamma_{new,2}$,$\Gamma_{new,3}$)
26
Path Metric Memory

Fig. 6

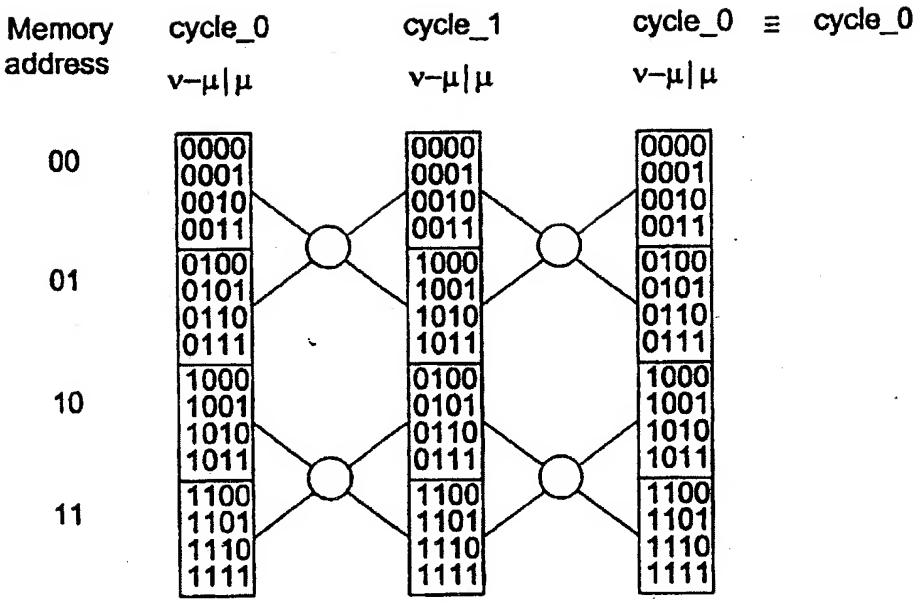

Memory address
cycle_0
ν–μ|μ
cycle_1
ν–μ|μ
cycle_0 ≡ cycle_0
ν–μ|μ
00
01
10
11
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

Fig. 7

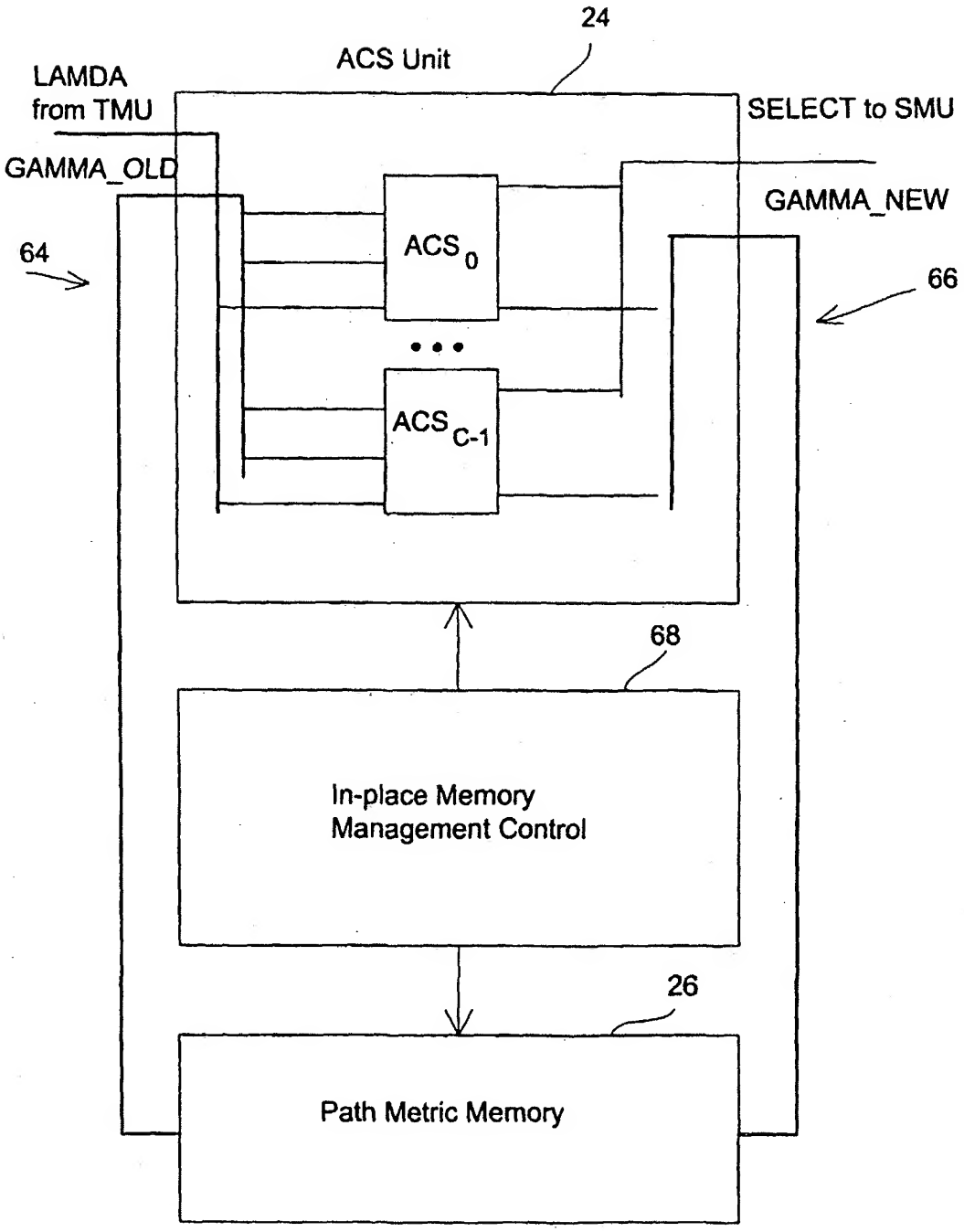

24
ACS Unit
LAMDA
from TMU
SELECT to SMU
GAMMA_OLD
GAMMA_NEW
64
66
ACS 0
ACS C-1
68
In-place Memory
Management Control
26
Path Metric Memory

Fig. 8

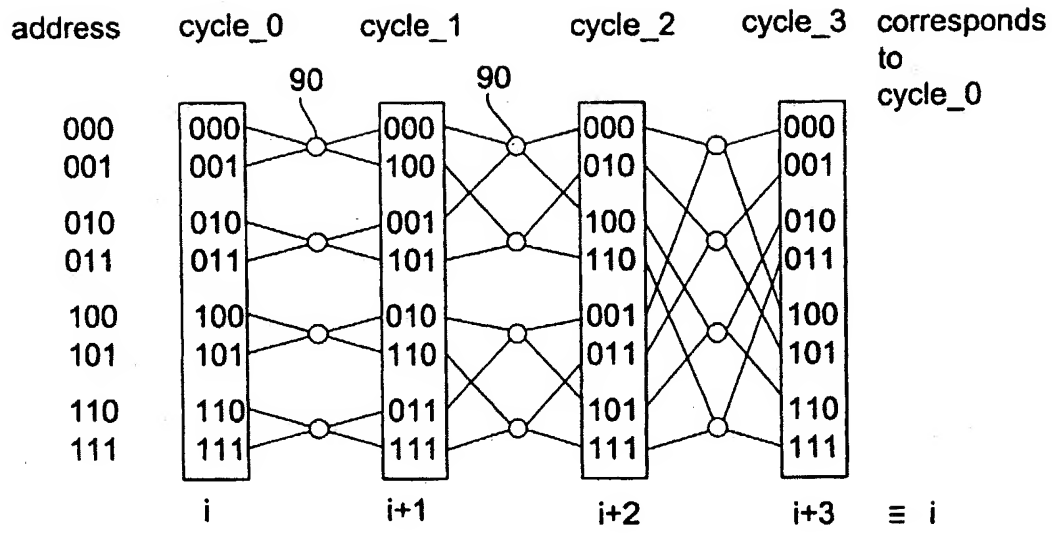


Fig. 9 (Prior Art)

INTERNATIONAL SEARCH REPORT

Interi nal Application No
PCT/EP 99/04725

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H03M13/23

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H03M

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	H. DAWID AND H. MEYR: "Scalable Architectures for High Speed Channel Decoding" VLSI SIGNAL PROCESSING VII, PROCEEDINGS OF 1994 IEEE WORKSHOP ON VLSI SIGNAL PROCESSING, 26 - 28 October 1994, pages 226-235, XP002132423 La Jolla, CA, USA page 226, line 1 -page 235, last line; figures 4,5 --- -/--	1

[X] Further documents are listed in the continuation of box C.

[ ] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
7 March 2000	29/03/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Van Staveren, M

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inte ional Application No
PCT/EP 99/04725

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	C.M. RADER: "Memory Management in a Viterbi Decoder" IEEE TRANSACTIONS ON COMMUNICATIONS, vol. com-29, no. 9, September 1981 (1981-09), pages 1399-1401, XP000877057 cited in the application page 1399, column 1, line 1 -page 1401, column 2, line 3; figure 2	1

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)